

EXERCICE 1

On donne la chaîne de caractères binaire = "11001001".

1. Que vaut `binaire[0]`?
"1"
2. Que vaut `binaire[-1]`?
"1"
3. Que vaut `binaire[2]*3 + binaire[4]*2`?
"00011"
4. On écrit dans un shell Python la commande suivante:

```
>>> binaire[3] = 3
```

Quel type de résultat est affiché après validation de cette instruction?

On obtient une erreur, car une chaîne de caractères est immuable. (TypeError: 'str' object does not support item assignment)

EXERCICE 2

Écrire une fonction `supprimer_occurrence` qui prend pour paramètre une chaîne de caractère `mot` et un caractère `c` et qui renvoie une chaîne où le caractère `c` a été supprimé de la chaîne `mot`.

On s'interdit d'utiliser une fonction prédéfinie du langage Python. Par exemple

```
>>> supprimer_occurrence("magie" , 'i')
"mage"
>>> supprimer_occurrence("tourterelle" , "t")
"ourerelle"
```

```
1 def supprimer_occurrence(mot,c):
2     nouveau = ""
3     for lettre in mot:
4         if lettre != c:
5             nouveau = nouveau + c
6     return nouveau
```

EXERCICE 3

On donne la fonction `mystere` ci-dessous qui prend pour paramètres deux nombres entiers `a` et `b` et qui renvoie un entier:

```
1 def mystere(a,b : int)->int:
2     u = 1
3     v = 1
4     for i in range(4):
5         w = u + v
6         v = u
7         u = w
8     return w
```

Que renvoie l'appel `mystere(1,1)`? Pour répondre à cette question , on complètera le tableau (dit tableau d'avancement) suivant :

fin de l'itération i	w	v	u
i = 0	2	1	2
i = 1	3		
i = 2			
i = 3			

Correction:

fin de l'itération i	w	v	u
i = 0	2	1	2
i = 1	3	2	3
i = 2	5	3	5
i = 3	8	5	8

`mystere(1,1)` vaut 8.

EXERCICE 4

On étudie dans cet exercice des robots qui se déplacent .

Chaque robot peut se déplacer en avant et tourner de 90 degrés à droite ou à gauche. Il reçoit ses instructions sous la forme d'une chaîne de caractères contenant des lettres et éventuellement des entiers .

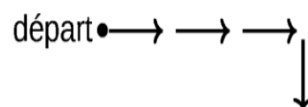
S'il lit un 'A', le robot avance d'une distance correspondant à "un pas" (valeur fixée dans les réglages du robot).

Il tourne à droite s'il lit un 'D' et à gauche s'il lit un 'G'. Insérer un entier dans la chaîne permet de répéter une action .

On suppose ici qu'un tel entier est compris entre 1 et 9.

'5A' est équivalent à 'AAAAA'.

Par exemple , la figure 1 représente le parcours du robot lorsqu'il reçoit la chaîne '3ADA'



1. Représenter le parcours effectué par le robot s'il reçoit la chaîne '4AGAA'.
2. On dit que la chaîne de caractères représentant le déplacement du robot est valide si les deux règles suivantes sont vérifiées:
 - Tous les caractères de la chaîne sont des caractères inclus dans '123456789ADG'
 - Aucun caractère représentant un entier ne peut être suivi par les lettres 'G' ou 'D' et que cet entier ne figure pas en fin de chaîne.

Écrire une fonction `est_valide` qui prend en paramètre une chaîne de caractères `chaîne` (représente le parcours du robot) et qui renvoie un booléen indiquant si la chaîne est valide.

On pourra recopier le code ci-dessous en complétant les lignes incomplètes.

```

1 def est_valide(chaine:str)->bool:
2     entiers = "123456789"
3     i = 0
4     while i < len(chaine):
5         if chaine[i] in entiers:
6             if i ==len(chaine):
7                 return False
8             elif i + 1 <len(chaine) and chaine[i+1] in "GD":
9                 return False
10            elif not chaine[i] in "AGD":
11                return False
12            i = i + 1
13        return True
14    ##quelques tests
15    assert est_valide("AA3AGA") == True
16    assert est_valide("AA6") == False
17    assert est_valide("3GAA") == False

```

3. La distance parcourue par un robot est calculée en nombre de pas (on rappelle que la lettre 'A' correspond à un pas).

Écrire une fonction `distance(chaine)` qui prend pour paramètre une chaîne de caractères décrivant le parcours d'un robot et qui renvoie la distance parcourue par le robot.

l'algorithme :

Pour calculer la distance ,on définit une variable d on parcourt la chaine de gauche à droite:

Si on lit un 'A' , on ajoute 1 à d

Si on lit un chiffre , on ajoute la

valeur de ce chiffre à d et on saute un rang

avant de continuer la lecture pour éviter de

compter un 'A' en trop

```

1 def distance(parcours :str)->int:
2     '''
3     parcours : une chaine valide
4     d : (int) la distance parcourue
5     '''
6     d = 0
7     i = 0
8     while i < len(parcours):
9         c = parcours[i]
10        #si on lit un 'A' non précédé par un chiffre
11        if c=='A':
12            d +=1
13        elif c in "12345678":
14            d += int(c)
15            i = i + 1 #on avance d'un rang dans la chaîne
16                    # pour éviter de compter un 'A' en trop
17        i = i + 1
18    return d

```

```
assert distance("AA3AGA") == 6
```

```
assert distance("AAAA2A5AGA") == 12
```

Une autre solution plus élégante proposée par un élève de la classe:

```
1 def distance(chaine):
2     dist = 0
3     entiers = "123456789"
4     for i in chaine:
5         if i == "A":
6             dist = dist + 1
7         elif i in entiers:
8             dist += int(i) - 1
9     return dist
```